**APPLE ITUNES MUSIC ANALYSIS PROJECT SQL.sql**

CREATE DATABASE MUSIC\_DATABASE;

USE MUSIC\_DATABASE;

CREATE TABLE ALBUMS\_DATASET(

ALBUM\_ID INTEGER PRIMARY KEY,

TITLE TEXT NOT NULL,

ARTIST INTEGER NOT NULL);

CREATE TABLE ARTIST\_DATA(

ARTIST\_ID INTEGER PRIMARY KEY,

NAME TEXT NOT NULL);

CREATE TABLE employee\_data (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(50) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

title VARCHAR(100),

reports\_to INT,

levels VARCHAR(10),

birthdate DATETIME,

hire\_date DATETIME,

address VARCHAR(255),

city VARCHAR(100),

state VARCHAR(50),

country VARCHAR(50),

postal\_code VARCHAR(20),

phone VARCHAR(50),

fax VARCHAR(50),

email VARCHAR(255)

);

CREATE TABLE genre (

genre\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

CREATE TABLE invoice (

invoice\_id INT PRIMARY KEY,

customer\_id INT NOT NULL,

invoice\_date DATETIME NOT NULL,

billing\_address VARCHAR(255),

billing\_city VARCHAR(100),

billing\_state VARCHAR(50),

billing\_country VARCHAR(50),

billing\_postal\_code VARCHAR(20),

total DECIMAL(10, 2) NOT NULL

);

CREATE TABLE invoice\_line (

invoice\_line\_id INTEGER PRIMARY KEY,

invoice\_id INTEGER,

track\_id INTEGER,

unit\_price DECIMAL(10, 2),

quantity INTEGER

);

CREATE TABLE media\_type (

media\_type\_id INTEGER PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

CREATE TABLE playlist (

playlist\_id INT PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

CREATE TABLE playlist\_track (

playlist\_id INT NOT NULL,

track\_id INT NOT NULL,

PRIMARY KEY (playlist\_id, track\_id),

FOREIGN KEY (playlist\_id) REFERENCES playlist(playlist\_id)

);

CREATE TABLE customer (

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(40) NOT NULL,

last\_name VARCHAR(40) NOT NULL,

company VARCHAR(80),

address VARCHAR(120),

city VARCHAR(60),

state VARCHAR(40),

country VARCHAR(40),

postal\_code VARCHAR(20),

phone VARCHAR(30),

fax VARCHAR(30),

email VARCHAR(80),

support\_rep\_id INT

);

CREATE TABLE track (

track\_id INTEGER PRIMARY KEY,

name VARCHAR(255),

album\_id INTEGER,

media\_type\_id INTEGER,

genre\_id INTEGER,

composer VARCHAR(255),

milliseconds INTEGER,

bytes INTEGER,

unit\_price NUMERIC(4,2)

);

select \* from employee\_data;

insert into employee\_data (employee\_id, last\_name, first\_name, title, reports\_to, levels,

birthdate, hire\_date, address, city, state, country, postal\_code, phone, fax, email) values

(1, 'Adams', 'Andrew','General Manager',9,'L6', '1962-02-18 00:00','2016-08-14 00:00','1120 Jasper Ave NW',

'Edmonton','AB','Canada','T5K 2N1','+1 (780)-428-9482','+1 (780)-428-3457','andrew@chinookcorp.com'),

(2,'Edwards','Nancy','Sales Manager',1,'L4', '1958-12-08 00:00','2016-05-01 00:00','825 8 Ave SW',

'Calgary','AB','Canada','T2P 2T3', '+1 (403) 262-3443', '+1 (403) 262-3322', 'nancy@chinookcorp.com'),

(3,'Peacock','Jane', 'Sales Support Agent',2, 'L1','1973-08-29 00:00','2017-04-01 00:00',

'1111 6 Ave SW', 'Calgary', 'AB', 'Canada', 'T2P 5M5', '+1 (403) 262-3443', '+1 (403) 262-6712', 'jane@chinookcorp.com'),

(4,'Park','Margaret','Sales Support Agent', 2,'L1', '1947-09-19 00:00', '2017-05-03 00:00',

'683 10 Street SW', 'Calgary', 'AB',' Canada', 'T2P 5G3',' +1 (403) 263-4423', '+1 (403) 263-4289', 'margaret@chinookcorp.com'),

(5,'Johnson','Steve', 'Sales Support Agent', 2, 'L1', '1965-03-03 00:00', '2017-10-17 00:00',

'7727B 41 Ave',' Calgary', 'AB','Canada', 'T3B 1Y7', '1 (780) 836-9987', '1 (780) 836-9543', 'steve@chinookcorp.com'),

(6,'Mitchell','Michael', 'IT Manager', 1, 'L3', '1973-07-01 00:00', '2016-10-17 00:00',

'5827 Bowness Road NW', 'Calgary', 'AB', 'Canada', 'T3B 0C5', '+1 (403) 246-9887', '+1 (403) 246-9899', 'michael@chinookcorp.com'),

(7, 'King', 'Robert', 'IT Staff',6, 'L2', '1970-05-29 00:00', '2017-01-02 00:00',

'590 Columbia Boulevard West', 'Lethbridge', 'AB', 'Canada', 'T1K 5N8', '+1 (403) 456-9986', '+1 (403) 456-8485', 'robert@chinookcorp.com'),

(8,'Callahan', 'Laura', 'IT Staff', 6, 'L2', '1968-01-09 00:00', '2017-03-04 00:00',

'923 7 ST NW', 'Lethbridge', 'AB', 'Canada', 'T1H 1Y8', '+1 (403) 467-3351', '+1 (403) 467-8772', 'laura@chinookcorp.com'),

(9, 'Madan', 'Mohan', 'Senior General Manager', 5,'L7', '1961-01-26 00:00', '2016-01-14 00:00',

'1008 Vrinda Ave MT', 'Edmonton', 'AB', 'Canada', 'T5K 2N1', '+1 (780) 428-9482', '+1 (780) 428-3457', 'madan.mohan@chinookcorp.com');

select \* from employee\_data;

select \* from albums\_dataset;

select \* from customer;

select \* from genre;

select \* from invoice;

select \* from invoice\_line;

select \* from media\_type;

select \* from playlist;

select \* from playlist\_track;

select \* from track;

**-- 1. Customer Overview**

SELECT

country,

COUNT(\*) as total\_customers,

ROUND(AVG(total), 2) as avg\_invoice\_value

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY country

ORDER BY total\_customers DESC;

**-- 2. Revenue Trends by Month**

SELECT

YEAR(invoice\_date) as year,

MONTH(invoice\_date) as month,

SUM(total) as monthly\_revenue,

ROUND(SUM(total) / SUM(SUM(total)) OVER (PARTITION BY YEAR(invoice\_date)) \* 100, 2) as revenue\_percentage

FROM invoice

GROUP BY YEAR(invoice\_date), MONTH(invoice\_date)

ORDER BY year, month;

**-- 3. Top Selling Artists**

SELECT

a.name as artist\_name,

COUNT(il.track\_id) as tracks\_sold,

SUM(il.unit\_price \* il.quantity) as total\_revenue

FROM ARTIST\_DATA a

JOIN ALBUMS\_DATASET al ON a.ARTIST\_ID = al.ARTIST -- Changed to al.ARTIST

JOIN track t ON al.ALBUM\_ID = t.album\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY a.name

ORDER BY total\_revenue DESC

LIMIT 10;

**-- 4. Customer Engagement by Support Rep**

SELECT

e.first\_name,

e.last\_name,

COUNT(DISTINCT c.customer\_id) as customers\_supported,

SUM(i.total) as total\_sales,

ROUND(AVG(i.total), 2) as avg\_sale\_value

FROM employee\_data e

JOIN customer c ON e.employee\_id = c.support\_rep\_id

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY e.employee\_id

ORDER BY total\_sales DESC;

**-- 5. Playlist Popularity**

SELECT

p.name as playlist\_name,

COUNT(pt.track\_id) as total\_tracks,

COUNT(DISTINCT il.invoice\_id) as times\_purchased

FROM playlist p

JOIN playlist\_track pt ON p.playlist\_id = pt.playlist\_id

LEFT JOIN invoice\_line il ON pt.track\_id = il.track\_id

GROUP BY p.playlist\_id

ORDER BY times\_purchased DESC;

**--- ADVANCED ANALYTICS ---**

-- 1. Customer Segmentation using RFM Analysis

WITH customer\_rfm AS (

SELECT

c.customer\_id,

c.first\_name,

c.last\_name,

DATEDIFF(MAX(i.invoice\_date), CURRENT\_DATE()) as recency,

COUNT(i.invoice\_id) as frequency,

SUM(i.total) as monetary,

NTILE(4) OVER (ORDER BY DATEDIFF(MAX(i.invoice\_date), CURRENT\_DATE()) DESC) as r\_score,

NTILE(4) OVER (ORDER BY COUNT(i.invoice\_id)) as f\_score,

NTILE(4) OVER (ORDER BY SUM(i.total)) as m\_score

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

)

SELECT

customer\_id,

first\_name,

last\_name,

recency,

frequency,

monetary,

r\_score,

f\_score,

m\_score,

CASE

WHEN r\_score = 4 AND f\_score >= 3 AND m\_score >= 3 THEN 'Champions'

WHEN r\_score >= 3 AND f\_score >= 3 THEN 'Loyal Customers'

WHEN r\_score >= 3 AND m\_score >= 3 THEN 'Potential Loyalists'

WHEN r\_score = 2 THEN 'Recent Customers'

WHEN r\_score = 1 THEN 'At Risk'

ELSE 'Need Attention'

END as customer\_segment

FROM customer\_rfm

ORDER BY monetary DESC;

**-- 2. Top Performing Tracks with Window Functions**

WITH track\_performance AS (

SELECT

t.track\_id,

t.name as track\_name,

a.name as artist\_name,

al.title as album\_name,

COUNT(il.invoice\_line\_id) as times\_purchased,

SUM(il.unit\_price \* il.quantity) as total\_revenue,

RANK() OVER (ORDER BY SUM(il.unit\_price \* il.quantity) DESC) as revenue\_rank,

DENSE\_RANK() OVER (PARTITION BY g.genre\_id ORDER BY SUM(il.unit\_price \* il.quantity) DESC) as genre\_rank

FROM track t

JOIN ALBUMS\_DATASET al ON t.album\_id = al.ALBUM\_ID

JOIN ARTIST\_DATA a ON al.ARTIST = a.ARTIST\_ID -- Changed to al.ARTIST

JOIN genre g ON t.genre\_id = g.genre\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY t.track\_id

)

SELECT

track\_name,

artist\_name,

album\_name,

times\_purchased,

total\_revenue,

revenue\_rank,

genre\_rank

FROM track\_performance

WHERE revenue\_rank <= 20

ORDER BY revenue\_rank;

**-- 3. Monthly Sales Growth Analysis**

WITH monthly\_sales AS (

SELECT

YEAR(invoice\_date) as year,

MONTH(invoice\_date) as month,

SUM(total) as monthly\_revenue,

LAG(SUM(total)) OVER (ORDER BY YEAR(invoice\_date), MONTH(invoice\_date)) as prev\_month\_revenue

FROM invoice

GROUP BY YEAR(invoice\_date), MONTH(invoice\_date)

)

SELECT

year,

month,

monthly\_revenue,

prev\_month\_revenue,

ROUND(((monthly\_revenue - prev\_month\_revenue) / prev\_month\_revenue) \* 100, 2) as growth\_percentage,

CASE

WHEN monthly\_revenue > prev\_month\_revenue THEN 'Growth'

WHEN monthly\_revenue < prev\_month\_revenue THEN 'Decline'

ELSE 'Stable'

END as trend

FROM monthly\_sales

ORDER BY year, month;

**-- 4. Customer Lifetime Value (CLV) Analysis**

WITH customer\_purchases AS (

SELECT

c.customer\_id,

c.first\_name,

c.last\_name,

COUNT(i.invoice\_id) as total\_purchases,

SUM(i.total) as total\_spent,

DATEDIFF(MAX(i.invoice\_date), MIN(i.invoice\_date)) as customer\_tenure\_days,

CASE

WHEN DATEDIFF(MAX(i.invoice\_date), MIN(i.invoice\_date)) = 0 THEN SUM(i.total)

ELSE SUM(i.total) / (DATEDIFF(MAX(i.invoice\_date), MIN(i.invoice\_date)) / 30.0)

END as monthly\_value

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

)

SELECT

customer\_id,

first\_name,

last\_name,

total\_purchases,

total\_spent,

customer\_tenure\_days,

monthly\_value,

NTILE(5) OVER (ORDER BY monthly\_value DESC) as value\_segment

FROM customer\_purchases

ORDER BY monthly\_value DESC;

**-- 5. Genre Popularity Over Time**

SELECT

g.name as genre\_name,

YEAR(i.invoice\_date) as year,

QUARTER(i.invoice\_date) as quarter,

COUNT(il.invoice\_line\_id) as tracks\_sold,

SUM(il.unit\_price \* il.quantity) as genre\_revenue,

ROUND(SUM(il.unit\_price \* il.quantity) / SUM(SUM(il.unit\_price \* il.quantity))

OVER (PARTITION BY YEAR(i.invoice\_date), QUARTER(i.invoice\_date)) \* 100, 2) as market\_share

FROM genre g

JOIN track t ON g.genre\_id = t.genre\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

JOIN invoice i ON il.invoice\_id = i.invoice\_id

GROUP BY g.genre\_id, YEAR(i.invoice\_date), QUARTER(i.invoice\_date)

ORDER BY year, quarter, genre\_revenue DESC;

**APPLE ITUNES MUSIC ANALYSIS PROJECT SQL\_1.sql**

**-- CUSTOMER ANALYTICS --**

**-- 1.1 Which customers have spent the most money on music?**

SELECT

c.customer\_id,

CONCAT(c.first\_name, ' ', c.last\_name) as customer\_name,

c.country,

c.email,

SUM(i.total) as total\_spent,

COUNT(i.invoice\_id) as total\_purchases

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

ORDER BY total\_spent DESC

LIMIT 10;

**-- 1.2 What is the average customer lifetime value?**

SELECT

ROUND(AVG(total\_spent), 2) as avg\_lifetime\_value,

ROUND(AVG(purchase\_count), 2) as avg\_purchases\_per\_customer

FROM (

SELECT

c.customer\_id,

SUM(i.total) as total\_spent,

COUNT(i.invoice\_id) as purchase\_count

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

) customer\_stats;

**-- 1.3 How many customers have made repeat purchases versus one-time purchases?**

SELECT

purchase\_type,

COUNT(\*) as customer\_count,

ROUND(COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM customer), 2) as percentage

FROM (

SELECT

c.customer\_id,

CASE

WHEN COUNT(i.invoice\_id) > 1 THEN 'Repeat Customer'

WHEN COUNT(i.invoice\_id) = 1 THEN 'One-time Customer'

ELSE 'No Purchase'

END as purchase\_type

FROM customer c

LEFT JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

) purchase\_categories

GROUP BY purchase\_type;

**-- 1.4 Which country generates the most revenue per customer?**

SELECT

country,

COUNT(DISTINCT c.customer\_id) as total\_customers,

SUM(i.total) as total\_revenue,

ROUND(SUM(i.total) / COUNT(DISTINCT c.customer\_id), 2) as revenue\_per\_customer

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY country

HAVING total\_customers > 5

ORDER BY revenue\_per\_customer DESC;

**-- 1.5 Which customers haven't made a purchase in the last 6 months?**

SELECT

c.customer\_id,

CONCAT(c.first\_name, ' ', c.last\_name) as customer\_name,

c.email,

c.country,

MAX(i.invoice\_date) as last\_purchase\_date,

DATEDIFF(CURDATE(), MAX(i.invoice\_date)) as days\_since\_last\_purchase

FROM customer c

LEFT JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

HAVING last\_purchase\_date IS NULL OR last\_purchase\_date < DATE\_SUB(CURDATE(), INTERVAL 6 MONTH)

ORDER BY days\_since\_last\_purchase DESC;

**-- SALES AND REVENUE ANALYSIS --**

**-- 2.1 Monthly revenue trends for the last two years**

SELECT

YEAR(invoice\_date) as year,

MONTH(invoice\_date) as month,

MONTHNAME(invoice\_date) as month\_name,

COUNT(invoice\_id) as total\_invoices,

SUM(total) as monthly\_revenue,

ROUND(AVG(total), 2) as avg\_invoice\_value

FROM invoice

WHERE invoice\_date >= DATE\_SUB((SELECT MAX(invoice\_date) FROM invoice), INTERVAL 2 YEAR)

GROUP BY YEAR(invoice\_date), MONTH(invoice\_date), MONTHNAME(invoice\_date)

ORDER BY year DESC, month DESC;

**-- 2.2 Average value of an invoice**

SELECT

ROUND(AVG(total), 2) as avg\_invoice\_value,

MIN(total) as min\_invoice\_value,

MAX(total) as max\_invoice\_value,

COUNT(\*) as total\_invoices

FROM invoice;

**-- 2.3 Payment methods analysis (assuming payment method is in invoice table)**

**-- If you have a payment\_method column, use this:**

SELECT

DAYNAME(invoice\_date) as day\_of\_week,

COUNT(\*) as transaction\_count,

SUM(total) as total\_revenue,

ROUND(AVG(total), 2) as avg\_transaction\_value,

ROUND(COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM invoice), 2) as percentage\_of\_total

FROM invoice

GROUP BY DAYNAME(invoice\_date), DAYOFWEEK(invoice\_date)

ORDER BY DAYOFWEEK(invoice\_date);

SELECT

HOUR(invoice\_date) as hour\_of\_day,

COUNT(\*) as transaction\_count,

SUM(total) as total\_revenue,

ROUND(AVG(total), 2) as avg\_transaction\_value

FROM invoice

GROUP BY HOUR(invoice\_date)

ORDER BY hour\_of\_day;

**-- 2.4 Revenue contribution by sales representative**

SELECT

e.employee\_id,

CONCAT(e.first\_name, ' ', e.last\_name) as sales\_rep,

e.title,

COUNT(DISTINCT c.customer\_id) as customers\_managed,

COUNT(i.invoice\_id) as total\_invoices,

SUM(i.total) as total\_revenue,

ROUND(SUM(i.total) / COUNT(i.invoice\_id), 2) as avg\_sale\_value

FROM employee\_data e

JOIN customer c ON e.employee\_id = c.support\_rep\_id

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY e.employee\_id

ORDER BY total\_revenue DESC;

**-- 2.5 Peak sales months or quarters**

SELECT

YEAR(invoice\_date) as year,

QUARTER(invoice\_date) as quarter,

COUNT(invoice\_id) as total\_invoices,

SUM(total) as quarterly\_revenue,

ROUND(SUM(total) / COUNT(DISTINCT MONTH(invoice\_date)), 2) as avg\_monthly\_revenue

FROM invoice

GROUP BY YEAR(invoice\_date), QUARTER(invoice\_date)

ORDER BY quarterly\_revenue DESC;

**-- PRODUCT & CONTENT ANALYSIS --**

**-- 3.1 Tracks that generated the most revenue**

SELECT

t.track\_id,

t.name as track\_name,

a.name as artist\_name,

al.title as album\_name,

g.name as genre,

COUNT(il.invoice\_line\_id) as times\_purchased,

SUM(il.unit\_price \* il.quantity) as total\_revenue

FROM track t

JOIN ALBUMS\_DATASET al ON t.album\_id = al.ALBUM\_ID

JOIN ARTIST\_DATA a ON al.ARTIST = a.ARTIST\_ID

JOIN genre g ON t.genre\_id = g.genre\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY t.track\_id

ORDER BY total\_revenue DESC

LIMIT 20;

**-- 3.2 Most frequently purchased albums**

SELECT

al.ALBUM\_ID,

al.title as album\_name,

a.name as artist\_name,

COUNT(DISTINCT il.invoice\_id) as times\_purchased,

COUNT(il.track\_id) as total\_tracks\_sold,

SUM(il.unit\_price \* il.quantity) as total\_revenue

FROM ALBUMS\_DATASET al

JOIN ARTIST\_DATA a ON al.ARTIST = a.ARTIST\_ID

JOIN track t ON al.ALBUM\_ID = t.album\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY al.ALBUM\_ID

ORDER BY times\_purchased DESC

LIMIT 15;

**-- 3.3 Tracks or albums that have never been purchased**

**-- Tracks never purchased**

SELECT

t.track\_id,

t.name as track\_name,

a.name as artist\_name,

al.title as album\_name

FROM track t

JOIN ALBUMS\_DATASET al ON t.album\_id = al.ALBUM\_ID

JOIN ARTIST\_DATA a ON al.ARTIST = a.ARTIST\_ID

LEFT JOIN invoice\_line il ON t.track\_id = il.track\_id

WHERE il.track\_id IS NULL;

**-- 3.4 Average price per track across different genres**

SELECT

g.genre\_id,

g.name as genre\_name,

COUNT(t.track\_id) as total\_tracks,

ROUND(AVG(t.unit\_price), 2) as avg\_track\_price,

SUM(il.unit\_price \* il.quantity) as total\_genre\_revenue,

COUNT(il.invoice\_line\_id) as total\_tracks\_sold

FROM genre g

LEFT JOIN track t ON g.genre\_id = t.genre\_id

LEFT JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY g.genre\_id

ORDER BY total\_genre\_revenue DESC;

**-- 3.5 Tracks per genre vs sales correlation**

SELECT

g.genre\_id,

g.name as genre\_name,

COUNT(DISTINCT t.track\_id) as available\_tracks,

COUNT(il.invoice\_line\_id) as tracks\_sold,

ROUND(COUNT(il.invoice\_line\_id) \* 100.0 / COUNT(DISTINCT t.track\_id), 2) as sales\_ratio,

SUM(il.unit\_price \* il.quantity) as total\_revenue

FROM genre g

LEFT JOIN track t ON g.genre\_id = t.genre\_id

LEFT JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY g.genre\_id

ORDER BY sales\_ratio DESC;

**APPLE ITUNES MUSIC ANALYSIS PROJECT SQL\_2.sql**

**-- ARTIST & GENRE PERFORMANCE --**

**-- 4.1 Top 5 highest-grossing artists**

SELECT

a.ARTIST\_ID,

a.name as artist\_name,

COUNT(DISTINCT al.ALBUM\_ID) as total\_albums,

COUNT(DISTINCT t.track\_id) as total\_tracks,

COUNT(il.invoice\_line\_id) as tracks\_sold,

SUM(il.unit\_price \* il.quantity) as total\_revenue

FROM ARTIST\_DATA a

JOIN ALBUMS\_DATASET al ON a.ARTIST\_ID = al.ARTIST

JOIN track t ON al.ALBUM\_ID = t.album\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY a.ARTIST\_ID

ORDER BY total\_revenue DESC

LIMIT 5;

**-- 4.2 Genre popularity by tracks sold**

SELECT

g.genre\_id,

g.name as genre\_name,

COUNT(il.invoice\_line\_id) as tracks\_sold,

SUM(il.unit\_price \* il.quantity) as total\_revenue,

ROUND(AVG(t.unit\_price), 2) as avg\_price

FROM genre g

JOIN track t ON g.genre\_id = t.genre\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY g.genre\_id

ORDER BY tracks\_sold DESC;

**-- 4.3 Genre popularity by revenue**

SELECT

g.genre\_id,

g.name as genre\_name,

SUM(il.unit\_price \* il.quantity) as total\_revenue,

COUNT(il.invoice\_line\_id) as tracks\_sold,

ROUND(SUM(il.unit\_price \* il.quantity) / COUNT(il.invoice\_line\_id), 2) as revenue\_per\_track

FROM genre g

JOIN track t ON g.genre\_id = t.genre\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY g.genre\_id

ORDER BY total\_revenue DESC;

**-- 4.4 Genre popularity by country**

SELECT

c.country,

g.name as genre\_name,

COUNT(il.invoice\_line\_id) as tracks\_sold,

SUM(il.unit\_price \* il.quantity) as total\_revenue

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

JOIN invoice\_line il ON i.invoice\_id = il.invoice\_id

JOIN track t ON il.track\_id = t.track\_id

JOIN genre g ON t.genre\_id = g.genre\_id

GROUP BY c.country, g.name

ORDER BY c.country, total\_revenue DESC;

**--- EMPLOYEE & OPERATIONAL EFFICIENCY ---**

**-- 5.1 Employees managing highest-spending customers**

SELECT

e.employee\_id,

CONCAT(e.first\_name, ' ', e.last\_name) as sales\_rep,

e.title,

COUNT(DISTINCT c.customer\_id) as total\_customers,

SUM(i.total) as total\_revenue,

ROUND(SUM(i.total) / COUNT(DISTINCT c.customer\_id), 2) as revenue\_per\_customer,

MAX(i.total) as largest\_sale

FROM employee\_data e

JOIN customer c ON e.employee\_id = c.support\_rep\_id

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY e.employee\_id

ORDER BY total\_revenue DESC;

**-- 5.2 Average number of customers per employee**

SELECT

ROUND(COUNT(DISTINCT c.customer\_id) \* 1.0 / COUNT(DISTINCT e.employee\_id), 2) as avg\_customers\_per\_rep

FROM employee\_data e

LEFT JOIN customer c ON e.employee\_id = c.support\_rep\_id

WHERE e.title LIKE '%Sales%' OR e.title LIKE '%Support%';

**-- 5.3 Revenue by employee regions**

SELECT

e.city as employee\_city,

e.country as employee\_country,

COUNT(DISTINCT e.employee\_id) as total\_employees,

COUNT(DISTINCT c.customer\_id) as customers\_managed,

SUM(i.total) as total\_revenue

FROM employee\_data e

JOIN customer c ON e.employee\_id = c.support\_rep\_id

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY e.city, e.country

ORDER BY total\_revenue DESC;

**-- GEOGRAPHIC TRENDS --**

**-- 6.1 Countries with highest number of customers**

SELECT

country,

COUNT(\*) as total\_customers,

SUM(i.total) as total\_revenue,

ROUND(SUM(i.total) / COUNT(\*), 2) as revenue\_per\_customer

FROM customer c

LEFT JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY country

ORDER BY total\_customers DESC;

**-- 6.2 Revenue variation by region**

SELECT

country,

COUNT(DISTINCT c.customer\_id) as total\_customers,

SUM(i.total) as total\_revenue,

ROUND(SUM(i.total) / COUNT(DISTINCT c.customer\_id), 2) as avg\_revenue\_per\_customer,

COUNT(i.invoice\_id) as total\_transactions

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY country

HAVING total\_customers >= 5

ORDER BY total\_revenue DESC;

**-- 6.3 Underserved geographic regions**

SELECT

country,

COUNT(\*) as total\_customers,

COALESCE(SUM(i.total), 0) as total\_revenue,

CASE

WHEN COUNT(\*) > 10 AND COALESCE(SUM(i.total), 0) < 100 THEN 'High Potential - Low Revenue'

WHEN COUNT(\*) > 5 AND COALESCE(SUM(i.total), 0) < 50 THEN 'Medium Potential - Low Revenue'

ELSE 'Adequately Served'

END as service\_status

FROM customer c

LEFT JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY country

ORDER BY total\_customers DESC;

**-- CUSTOMER RETENTION & PURCHASE RETURNS --**

**-- 7.1 Purchase frequency distribution**

SELECT

purchase\_frequency,

COUNT(\*) as customer\_count,

ROUND(COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM customer), 2) as percentage

FROM (

SELECT

c.customer\_id,

COUNT(i.invoice\_id) as purchase\_frequency

FROM customer c

LEFT JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

) freq\_table

GROUP BY purchase\_frequency

ORDER BY purchase\_frequency;

**-- 7.2 Average time between customer purchases**

SELECT

c.customer\_id,

CONCAT(c.first\_name, ' ', c.last\_name) as customer\_name,

COUNT(i.invoice\_id) as total\_purchases,

ROUND(DATEDIFF(MAX(i.invoice\_date), MIN(i.invoice\_date)) / NULLIF(COUNT(i.invoice\_id) - 1, 0), 2) as avg\_days\_between\_purchases

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.customer\_id

HAVING total\_purchases > 1

ORDER BY avg\_days\_between\_purchases;

**-- 7.3 Customers purchasing from multiple genres**

SELECT

multi\_genre\_customers,

COUNT(\*) as customer\_count,

ROUND(COUNT(\*) \* 100.0 / MAX(total\_customers), 2) as percentage

FROM (

SELECT

c.customer\_id,

COUNT(DISTINCT t.genre\_id) as genres\_purchased,

CASE

WHEN COUNT(DISTINCT t.genre\_id) > 1 THEN 'Multi-Genre Buyer'

WHEN COUNT(DISTINCT t.genre\_id) = 1 THEN 'Single-Genre Buyer'

ELSE 'No Purchase'

END as multi\_genre\_customers,

(SELECT COUNT(\*) FROM customer) as total\_customers

FROM customer c

LEFT JOIN invoice i ON c.customer\_id = i.customer\_id

LEFT JOIN invoice\_line il ON i.invoice\_id = il.invoice\_id

LEFT JOIN track t ON il.track\_id = t.track\_id

GROUP BY c.customer\_id

) genre\_analysis

GROUP BY multi\_genre\_customers;

**-- OPERATIONAL OPTIMIZATION --**

**-- 8.1 Common track combinations (purchased together)**

SELECT

t1.track\_id as track1\_id,

t1.name as track1\_name,

t2.track\_id as track2\_id,

t2.name as track2\_name,

COUNT(\*) as times\_purchased\_together

FROM invoice\_line il1

JOIN invoice\_line il2 ON il1.invoice\_id = il2.invoice\_id AND il1.track\_id < il2.track\_id

JOIN track t1 ON il1.track\_id = t1.track\_id

JOIN track t2 ON il2.track\_id = t2.track\_id

GROUP BY t1.track\_id, t2.track\_id

HAVING times\_purchased\_together >= 3

ORDER BY times\_purchased\_together DESC

LIMIT 20;

**-- 8.2 Pricing patterns and sales performance**

SELECT

price\_range,

COUNT(t.track\_id) as total\_tracks,

SUM(il.quantity) as total\_units\_sold,

ROUND(SUM(il.unit\_price \* il.quantity), 2) as total\_revenue,

ROUND(SUM(il.quantity) \* 1.0 / COUNT(t.track\_id), 2) as avg\_units\_per\_track

FROM (

SELECT

track\_id,

CASE

WHEN unit\_price < 0.50 THEN 'Under $0.50'

WHEN unit\_price < 0.75 THEN '$0.50-$0.74'

WHEN unit\_price < 1.00 THEN '$0.75-$0.99'

WHEN unit\_price < 1.25 THEN '$1.00-$1.24'

ELSE '$1.25+'

END as price\_range

FROM track

) price\_categories

JOIN track t ON price\_categories.track\_id = t.track\_id

LEFT JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY price\_range

ORDER BY total\_revenue DESC;

**-- 8.3 Media type usage trends**

SELECT

m.media\_type\_id,

m.name as media\_type,

YEAR(i.invoice\_date) as year,

COUNT(il.invoice\_line\_id) as tracks\_sold,

SUM(il.unit\_price \* il.quantity) as total\_revenue,

ROUND((COUNT(il.invoice\_line\_id) - LAG(COUNT(il.invoice\_line\_id))

OVER (PARTITION BY m.media\_type\_id ORDER BY YEAR(i.invoice\_date))) \* 100.0 /

LAG(COUNT(il.invoice\_line\_id)) OVER (PARTITION BY m.media\_type\_id ORDER BY YEAR(i.invoice\_date)), 2) as growth\_percentage

FROM media\_type m

JOIN track t ON m.media\_type\_id = t.media\_type\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

JOIN invoice i ON il.invoice\_id = i.invoice\_id

GROUP BY m.media\_type\_id, YEAR(i.invoice\_date)

ORDER BY m.media\_type\_id, year;

**APPLE ITUNES MUSIC ANALYSIS PROJECT SQL\_3sql**

**-- Who is the senior most employee based on job title? --**

SELECT \* FROM employee\_data ORDER BY levels DESC LIMIT 1;

**-- Which countries have the most Invoices? --**

SELECT billing\_country, COUNT(\*) AS invoice\_count FROM invoice GROUP BY billing\_country

ORDER BY invoice\_count DESC;

**-- What are top 3 values of total invoice? --**

SELECT total FROM invoice ORDER BY total DESC LIMIT 3;

**-- Which city has the best customers? --**

SELECT billing\_city, SUM(total) AS total\_revenue FROM invoice GROUP BY billing\_city

ORDER BY total\_revenue DESC LIMIT 1;

**-- Who is the best customer? --**

SELECT c.customer\_id, c.first\_name, c.last\_name, SUM(i.total) AS total\_spent FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id GROUP BY c.customer\_id ORDER BY total\_spent DESC

LIMIT 1;

**-- Rock Music listeners --**

SELECT DISTINCT c.email, c.first\_name, c.last\_name FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

JOIN invoice\_line il ON i.invoice\_id = il.invoice\_id JOIN track t ON il.track\_id = t.track\_id

JOIN genre g ON t.genre\_id = g.genre\_id WHERE g.name = 'Rock' ORDER BY c.email;

**-- Top 10 rock artists --**

SELECT a.name AS artist\_name, COUNT(t.track\_id) AS track\_count FROM ARTIST\_DATA a

JOIN ALBUMS\_DATASET al ON a.ARTIST\_ID = al.ARTIST JOIN track t ON al.ALBUM\_ID = t.album\_id

JOIN genre g ON t.genre\_id = g.genre\_id WHERE g.name = 'Rock' GROUP BY a.ARTIST\_ID

ORDER BY track\_count DESC LIMIT 10;

**-- Tracks longer than average length --**

SELECT name, milliseconds FROM track WHERE milliseconds > (SELECT AVG(milliseconds) FROM track)

ORDER BY milliseconds DESC;

**-- Amount spent by each customer on artists --**

SELECT c.first\_name, c.last\_name, a.name AS artist\_name,

SUM(il.unit\_price \* il.quantity) AS total\_spent FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

JOIN invoice\_line il ON i.invoice\_id = il.invoice\_id

JOIN track t ON il.track\_id = t.track\_id

JOIN ALBUMS\_DATASET al ON t.album\_id = al.ALBUM\_ID

JOIN ARTIST\_DATA a ON al.ARTIST = a.ARTIST\_ID

GROUP BY c.customer\_id, a.ARTIST\_ID

ORDER BY total\_spent DESC;

**-- Most popular music genre for each country --**

WITH country\_genre\_sales AS (

SELECT

i.billing\_country AS country,

g.name AS genre\_name,

COUNT(il.invoice\_line\_id) AS purchase\_count,

RANK() OVER (PARTITION BY i.billing\_country ORDER BY COUNT(il.invoice\_line\_id) DESC) AS rank\_num

FROM invoice i

JOIN invoice\_line il ON i.invoice\_id = il.invoice\_id

JOIN track t ON il.track\_id = t.track\_id

JOIN genre g ON t.genre\_id = g.genre\_id

GROUP BY i.billing\_country, g.genre\_id

)

SELECT country, genre\_name, purchase\_count

FROM country\_genre\_sales

WHERE rank\_num = 1

ORDER BY country;

**-- Top spending customer for each country --**

WITH customer\_spending AS (

SELECT

c.country,

c.customer\_id,

c.first\_name,

c.last\_name,

SUM(i.total) AS total\_spent,

RANK() OVER (PARTITION BY c.country ORDER BY SUM(i.total) DESC) AS rank\_num

FROM customer c

JOIN invoice i ON c.customer\_id = i.customer\_id

GROUP BY c.country, c.customer\_id

)

SELECT country, first\_name, last\_name, total\_spent

FROM customer\_spending

WHERE rank\_num = 1

ORDER BY country;

**-- Most popular artists --**

SELECT

a.name AS artist\_name,

COUNT(il.invoice\_line\_id) AS tracks\_sold,

SUM(il.unit\_price \* il.quantity) AS total\_revenue

FROM ARTIST\_DATA a

JOIN ALBUMS\_DATASET al ON a.ARTIST\_ID = al.ARTIST

JOIN track t ON al.ALBUM\_ID = t.album\_id

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY a.ARTIST\_ID

ORDER BY tracks\_sold DESC

LIMIT 10;

**-- Most popular song --**

SELECT

t.name AS track\_name,

a.name AS artist\_name,

COUNT(il.invoice\_line\_id) AS times\_purchased,

SUM(il.unit\_price \* il.quantity) AS total\_revenue

FROM track t

JOIN ALBUMS\_DATASET al ON t.album\_id = al.ALBUM\_ID

JOIN ARTIST\_DATA a ON al.ARTIST = a.ARTIST\_ID

JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY t.track\_id

ORDER BY times\_purchased DESC

LIMIT 1;

**-- Average prices of different music types --**

SELECT

g.name AS genre,

ROUND(AVG(t.unit\_price), 2) AS avg\_price,

COUNT(t.track\_id) AS total\_tracks,

SUM(il.unit\_price \* il.quantity) AS total\_revenue

FROM genre g

JOIN track t ON g.genre\_id = t.genre\_id

LEFT JOIN invoice\_line il ON t.track\_id = il.track\_id

GROUP BY g.genre\_id

ORDER BY avg\_price DESC;

**-- Most popular countries for music purchases --**

SELECT

billing\_country AS country,

COUNT(\*) AS total\_invoices,

SUM(total) AS total\_revenue,

ROUND(AVG(total), 2) AS avg\_invoice\_value,

COUNT(DISTINCT customer\_id) AS unique\_customers

FROM invoice

GROUP BY billing\_country

ORDER BY total\_revenue DESC;